Hydra: An Open Framework for Virtual-Fusion of Recommendation Filters

Savvas Karagiannidis, Stefanos Antaris, Christos Zigkolis, Athena Vakali
Department of Informatics
Aristotle University, 54124
Thessaloniki, Greece
{sakaragi, santaris, chzigkol, avakali}@csd.auth.gr

ABSTRACT

Today's web commercial applications demand more powerful recommendation systems due to the rapid increase in the number of both consumers and available products. Searching for the best algorithm with the highest accuracy and realistic complexity is, most of the time, a very costly process in terms of both time and resources. In this paper we suggest an alternative framework called Hydra which enables the virtual fusion of any and as many currently available recommendation algorithms in such a distributed manner that algorithms' complexities are not summarized but parallelized. Therefore, we utilize the available algorithms and technologies aiming to achieve better accuracy in order to surpass even the most state of the art algorithms. In addition, Hydra can be used to find how algorithms interact with each other in order to estimate the resulting accuracy towards inventing a more precise algorithm diminishing the risk of a failed investment. Hydra can be adjusted and integrated in any recommendation application while it is also open to new functionalities which can be embedded easily and in a transparent manner.

Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Decision support; H.3.3 [**Information Search and Retrieval**]: Information filtering; D.1.3 [**Concurrent Programming**]: Parallel programming

General Terms

Algorithms,Design,Performance,Theory

Keywords

Recommendation system, Framework, recommender, dynamic, modular, virtual fusion, algorithms combination, multiple algorithms, parallel system, adaptable, open, expansible

RecSys2010, September 26–30, 2010, Barcelona, Spain.

1. INTRODUCTION

Enterprises in our days (e.g. Amazon, Netflix) invest great revenues for the creation of both novel recommendation algorithms and systems aiming at an increased accuracy, for the purpose of enhancing the quality of their services. Even with these great investments there is the *risk of failure* due to a better algorithm and system from a competitor or of inferior results from the expected ones.

Another crucial issue in developing recommendation algorithms is their complexity. Hybrid algorithms [2] are the most effective ones in terms of accuracy. However, most of them are not parallelizable and this fact makes them, many times, too complex since two or more algorithms will need to be run consecutively and combine their results. As a result most Hybrid algorithms are inappropriate for online recommendation with large scale data [5].

Finally, apart from the algorithmic perspective, the system upon which the algorithm would be based is important. New functionalities and new items to be recommended might need to be added as time passes, and perhaps the effective algorithm replaced in favour of a new one. These would bring about many changes and heavy modification at a static system.

In order to solve the above problems, the Hydra Open Framework was developed. Hydra is an **open** and **modular framework** that aims to create **expansible** and **dynamic** recommendation systems in a **parallelized** and **distributed** manner, utilizing a mechanism (**Virtual Fusion**) for combining different and multiple algorithms together. Below we describe some of its stronger points in terms of content, architecture and complexity in short while further details will be given at a later section.

- **Content Independent**. Hydra can be easily adjusted to recommend any kind of items and therefore it can be easily moved from one application to another without any code modification.
- **Open Architecture**. All of Hydra's main functions are split into modules that run in parallel taking advantage of today's parallel systems. Additionally new functions may be added to the framework by adding new modules to the system and tuning the framework for the new module. Also new algorithms can be added and existent removed without any code change but by mere modification of the external parameters. The framework will run all available algorithms in parallel and through a **virtual-fusion** module combine their results to a final top-k set. This way one may also

divide the overall computational time to the system's processors. Hydra is also **cross-platform** taking advantage of java technologies and tools which are made to and tested in all modern computer systems and OS's. Finally, Hydra is developed to be **scalable** as it is created to run in distributed systems, with minimum communication needs between the modules and thus between the connected systems. This way the framework responds better than centralized unmodular systems at the increase of data and computational needs. We will further discuss those characteristics in the Framework Description section of this paper.

- **Complexity Distribution**. Hybrid algorithms are capable of increasing accuracy through combination of algorithms. However most of those are not parallelizable bringing about additional complexity. Research is being carried out to reduce complexity but most of the time it is swapped from the online to the offline complexity [5]. Hydra is being created on the idea of the **virtual-fusion** of multiple recommendation filters. Instead of creating a new hybrid algorithm that combines a number of algorithms and greatly increase complexity, any number of algorithms are being virtually fused by having them run in parallel and combining their results through a conflict resolution **policy**. We shall discuss the policy that is applied in this framework further, although once more Hydra is open to any new policy one wishes to have. With this virtual-fusion, in overall, the recommendation process will be as slow as the slowest of the algorithms that run on the framework. Finally, with the ability of virtual-fusion, a preliminary research can be carried out to find through virtual-fusion which recommendation techniques combine best.

2. FRAMEWORK

This section includes a description of the Hydra Framework, its current modules and functionalities. A general schema will be presented in order to better understand where Hydra is positioned, which modules communicate and how. Nonetheless, Hydra can be further expanded with the addition of new functionalities wherever required.

2.1 Where is Hydra positioned?

Hydra, just like any Recommendation System, is positioned between the system's hardware and the portal. Hydra wraps around a DBMS that is required for us to store data, retrieve data and extract information through the recommendation algorithms that we use on our application. The DBMS also handles certain XML files that provide the parameters required by the various modules of Hydra. Only those XML files need **tuning** whenever an additional algorithm or an additional data source is added to our framework.

2.2 The parts of Hydra

Hydra in its current form comprises of five main modules. All five are required for the Framework to operate as a Recommendation System, to **virtually-fuse** Recommendation Algorithms, to tune the parameters of the Framework instead of changing the code and to retrieve data from various data sources utilizing different technologies to serve requests. Figure 1 illustrates Hydra's general architecture and its components. The red single dot dashed boxes indicate the modules of Hydra, the green double dot dashed ones the parameters pools that are necessary for some modules to function and change at the will of the administrator, the blue dotted ones indicate modules of Hydra that are dynamically created during the operation of Hydra and finally the black boxes indicate independent cooperating systems.

- **Formatting Module-FM**

  FM is responsible for retrieving and storing data from available data sources, through use of their APIs, into Hydra's DB in the appropriate fields. Apart from retrieving new data, FM is responsible for updating the stored data synchronizing the local data with the external. FM has been developed to be able to take several roles and to call itself for the various procedures. At the startup of Hydra, a master FM is being loaded and executed in one thread, providing the administrator with monitoring activity, and copies of FM are created in different threads to carry out data retrieval or data update procedures. This way we divide the computational time of executing these procedures to the number of threads created. The parameters required for each of the available APIs and data sources are defined in an external XML file from where they can be changed due to addition or removal of an API.

- **Dataset Retrieval Module-DRM** DRM is responsible for choosing an appropriate subset of the DB to pass to each algorithm. Although a random policy could be easily chosen, this module was developed to make an initial preprocessing. It is responsible for creating an initial query taking into consideration the preferences previously recorded for the user. This query then will be used to measure the relevance of the available records, by use of a binary model, ranking them in descending order. The top-m records will be chosen as input data for the algorithms. In other words with this module we use a search-like process commonly used in Information Retrieval processes to create a dataset more relevant to the user's preferences than that of a random policy. DRM is the first process of the top-k recommendations extraction process.

- **Information Extraction Module-IEM** IEM is responsible for the recommendation of top-$n * k$ items, where n is the number of algorithms, for each of the users, by use of an offline recommendation process. This is achieved by running each and every available to the system recommendation algorithm in parallel and appending their results.IEM itself communicates with two other modules, the Dataset Retrieval Module and the Results Combination Module. Both are essential for an efficient and accurate item recommendation process. The operations of each of the two will be discussed below. IEM operates on input from DRM creating the input for RCM. It is the intermediate process of extracting top-k recommendations for each of the users. This intermediate process includes the startup of each of the algorithms, the insertion of each in a different thread and the passing of the required dataset to each of the algorithms. IEM is the second process of the top-k recommendations extraction process.

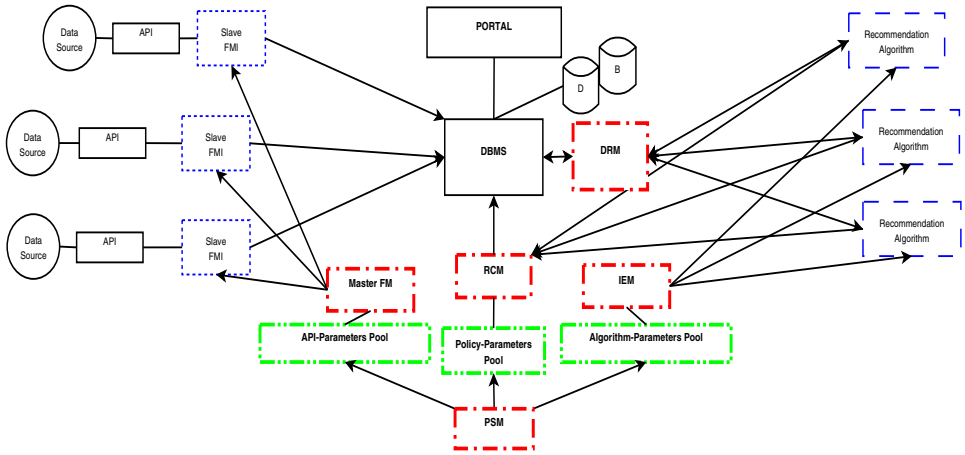

Figure 1: Hydra's Architecture

- **Results Combination Module-RCM**

  This module is responsible for the actual **virtual-fusion** in Hydra. With this module, Hydra does not need to combine algorithms but just results. With the use of a certain policy the top-$n*k$ results are combined and top-k results are extracted. A policy that is currently inserted in Hydra will be discussed in detail in the experiment section of the paper. RCM is the final process of the top-k recommendations extraction process.

- **Parameters Setup Module-PSM**

  This module was created for the purpose of easily changing the parameters of FM, RCM and IEM. Through this module, the administrator may add new data sources, clarify whether they work with http request or rpc etc. Also through this module the administrator can add new algorithms, remove existing ones, set the path of the algorithm and set which fields will be accepted as input for each of the algorithms. Finally for RCM, this module provides a way for the administrator to change the policy we enforce on the virtual-fusion of the algorithms' results.

3. EXPERIMENTATION

For the purpose of testing our system, a short length first experiment was carried out. We tuned the framework to work on books recommendation. We retrieved 65.000 books from 152 categories from ISBN Database [1] and worked on a sample of 25 users. The experiment would be divided into 3 stages.

3.1 Algorithms used

Content-Based Filtering-CBF:A CB algorithm was created for addition to Hydra's algorithms pool. The algorithm is actually a two field correlator which was used in the experiment to correlate the users' category preferences with the items categories and provide a weight according to the degree of correlation for each item. The algorithm is simple yet effective but more appropriate for small datasets.

Collaborative Filtering-CF:A slope-one recommender was used here [3]. Although simple, it has proven to provide results similar to those of more complex algorithms.

Hybrid Filtering-HF:A weighted hybrid algorithm was used combining lineary a slope-one recommender and a Pearson User Similarity algorithm [4].

Virtual Fusion-VF: Virtual Fusion is the process of combining the results from each and every algorithm in our system. The current policy is a combination of the weighted hybrid and mixed hybrid methodologies which are parallelizable. The policy currently used has the following steps:

1. Run all algorithms in seperate threads
2. Gather the results which must include a produced weight
3. Assign normalised to one weights to each algorithm according to previously calculated accuracy
4. Calculate overall weight of each item, multiplying the algorithm weight with each items's produced weight
5. Extract top-k items from the top-$n*k$, where n : number of algorithms, by summing up their Overall weight if there is a common recommendation and taking the k items with the greatest overall weight.

Complexity for each user:$max(ACi)+2*(m*k)+[((m*k)*log(m*k))+(m*k)]$, where AC is Algorithm Complexity, k is the number of recommended items and m the number of algorithms. We take the max complexity of all the Algorithms since all algorithms are run in parallel under the assumption, ofcourse, that there are at least so many available processors for **VF** as the number of algorithms used in the recommendation process. Also, $2*(m*k)$ refers to the 3 and 4 steps and the last part is the sorting of the new top-$n*k$ items and merging of the common items. Since $k << n$, where n is the number of items, used in the algorithms it can be easily said that the complexity of **VF** equals that of the most complex algorithm used. It is not our objective to show the exact complexity of all the algorithms, but to show that **VF** does not increase it.

The assumption of at least as many processors as algorithms is true for a small number of algorithms since contemporary MIMD systems have a large number of available processors. When the number of algorithms is scaled up in a single parallel system the number of processors could be smaller. The latter problem could be compensated when we

operate in a distributed environment which brings the problem of communication, fortunately required only between the algorithms and the DBMS in our system.

3.2 Experiment Stages

In this subsection, the stages of the experiment carried out are presented to get an idea of the system's algorithms overall accuracy in a system cold start stage.

1. **Registration**: Each user was requested to register into the portal of Hydra entering their personal data and category preferences. This was done in order to avoid the system cold-start problem and new user-ramp up problem[2].

2. **Content Based Filter Stage**: For the second part, 10 recommendations would be given to each user through use of the system's CBF. Also 40 randomly chosen books from among a 9.400 books subset would be given for rating in order to have some additional rates for the collaborative algorithms to operate.

3. **Collaborative-Hybrid Filter Stage**: For the third part, 10 recommendations would be provided from the previously shown Collaborative Filter and an additional ten from the Hybrid algorithm.

Finally, for the VF, no ratings needed to be given from the users since the CBF stage ratings were used as input for objectiveness and all the algorithms had provided recommendations which were already rated in previous stages.

We ran the above experimentation steps and came up with the results (see Figure 2) that shows that Virtual Fusion achieves similar accuracy to the Hybrid's one, also it is shown that the CBF provides increased accuracy compared to normally being by far surpassed by collaborative or hybrid algorithms. In Figure 2 we divided our users into groups of five and the average accuracies of each algorithm are presented, as can be seen, in different grouped bars. In such an early stage and with such few ratings data, we cannot reach a conclusion as to how beneficial Virtual Fusion is, however as we previously said, the policy used for Virtual Fusion at this point bears much better results as ratings increase [2].

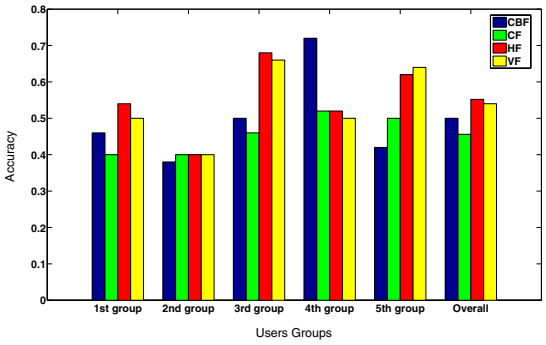

Figure 2: Algorithms' Accuracy Results

4. HYDRA USE SCENARIOS

Hydra can be used on a large number of research applications other than just in enteprise applications.

- **Optimized filter combination**. Finding and establishing the best combination of algorithms for a certain application is always an important issue on the field of Recommendation Systems. Hydra simplifies and accelerates this combination process as no new system needs to be created. Also, by using the policies of Hydra for conflict resolution, one can test which policy gives better accuracy.

- **Creation of new Hybrid algorithms**. With Hydra, a researcher can easily find which available algorithms and through which hybrid technique combine better since the latter can be represented by a VF policy. This way the creation of the Hybrid algorithm can be done without the risk of inferior than expected results.

- **Finding the critical dimension**. Although an item may consist of infinite fields, the values of a subset of those will affect the user's preferences. Finding the field that affects the preferences of the user the most, through targeting each field with the set of algorithms, we can find the critical dimension. With this knowledge the accuracy of the system can be improved.

5. CONCLUSION AND FUTURE WORK

In this paper an open framework was presented capable of adjusting to any recommendation based application, using the idea of virtual fusion, a way of enforcing a policy of combining the algorithms' outputs. The policy used in the experiments is a combination of the mixed and weighted hybrid methodologies, providing a parallelizable combination method. As our future work we intend on carrying out more experiments to show the increase of accuracy by the current **VF** methodology and further expand Hydra with the addition of more modules for preprocessing of datasets, insertion of more algorithms into Hydra's algorithms pool, creation of more VF policies and experimentation on other datasets to find the most critical datafield and most effective algorithm for each application.

6. REFERENCES

[1] International Standard Book Number (ISBN) Database, http://isbndb.com.

[2] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[3] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, 2005.

[4] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM.

[5] C. Shahabi and Y.-S. Chen. An adaptive recommendation system without explicit acquisition of user relevance feedback. *Distrib. Parallel Databases*, 14(2):173–192, 2003.